Nola Local- API Specifications

API Overview

- **Base URL**: `/api`
- **Architecture**: RESTful API
- **Authentication**: JWT Bearer tokens in Authorization header
- **Response Format**: JSON
- **Error Handling**: Consistent error response structure across all endpoints

---

1. Authentication Routes

GET `/api/auth/status`

Purpose: Check if user is authenticated (for client-side auth state)

- **Auth Required**: No

Success Response (200):
{ "isAuthenticated": true, "user": { "id": "...", "username": "..." } }

Success Response (200):
{ "isAuthenticated": false }

POST `/api/auth/signup`

Purpose: Register new user account

- **Auth Required**: No
- **Request Body**: `{ username, email, password }`

Success Response (201):
{ "message": "User created. Check email to verify account.", "userId": "ObjectId"}

- **Error Response (400)**: Validation errors (missing fields, invalid email format, password too weak)
- **Error Response (409)**: Username or email already exists
- **Side Effects**:
    - Creates user with `isVerified: false`

- Generates verification token
- Sends verification email via Nodemailer

POST `/api/auth/login`

Purpose: Authenticate user and receive JWT

- **Auth Required**: No
- **Request Body**: `{ email, password }`

Success Response (200):
{ "token": "JWT_STRING", "user": { "id": "...", "username": "...", "email": "...", "isVerified": true }}

Error Response (400): Missing credentials

- **Error Response (401)**: Invalid email or password
- **Error Response (403)**: Email not verified - prompt to check email or resend verification

---

POST `/api/auth/verifyemail`

Purpose: Verify user email with token from email link

- **Auth Required**: No
- **Request Body**: `{ token }`

Success Response (200):
{ "message": "Email verified successfully. You can now log in." }

- **Error Response (400)**: Token missing from request
- **Error Response (404)**: Invalid or expired token
- **Side Effects**:
  - Sets `isVerified: true`
  - Clears `verifyToken` and `verifyTokenExpiry` fields

POST `/api/auth/resend-verification`

Purpose: Resend verification email

- **Auth Required**: No

- **Request Body**: `{ email }`

Success Response (200):
{ "message": "Verification email sent successfully." }

- **Error Response (404)**: Email not found in system
- **Error Response (400)**: Account already verified

POST `/api/auth/logout`

Purpose: Log out user (primarily client-side token removal, optional server-side blacklisting)

- **Auth Required**: Yes
- **Request Body**: None

Success Response (200):
{ "message": "Logged out successfully" }

- **Note**: Frontend clears JWT from localStorage/Redux

---

2. Events Routes

GET `/api/events`

Purpose: Get all events with optional filters and search

- **Auth Required**: Yes
- **Query Parameters** (all optional):
    - `category`: string (category slug, e.g., "live-music")
    - `startDate`: ISO date string (events on/after this date)
    - `endDate`: ISO date string (events on/before this date)
    - `source`: string ("user" | "eventbrite" | "ticketmaster" | "all")
    - `search`: string (search in title and description)
    - `status`: string ("upcoming" | "passed", defaults to "upcoming")
    - `limit`: number (default: 50, max: 100)
    - `skip`: number (for pagination, default: 0)

- **Example**: `/api/events?category=live-music&startDate=2025-10-01&search=jazz&limit=20`

Success Response (200):
{ "events": [{ "_id": "ObjectId", "title": "Jazz Night", "description": "...", "date": "2025-10-15T00:00:00.000Z", "time": "8:00 PM", "location": "Frenchmen Street", "category": { "_id": "...", "name": "Live Music", "slug": "live-music", "color": "#FF6B6B" }, "imageUrl": "https://...", "source": "user", "sourceUrl": null, "creator": { "_id": "...", "username": "jazzfan123" }, "likesCount": 15, "isLikedByUser": false, "isCreator": false, "status": "upcoming" }], "total": 45, "hasMore": true}

- **Error Response (401)**: Not authenticated - redirect to login
- **Note**: Returns events with populated category and creator info

---

GET `/api/events/featured`

Purpose: Get featured events for public landing page carousel

- **Auth Required**: No (public endpoint)
- **Query Parameters**: None

Success Response (200):
{ "events": [{ "_id": "ObjectId", "title": "Mardi Gras Parade", "imageUrl": "https://...", "date": "2025-02-25T00:00:00.000Z", "category": "Community", "location": "St. Charles Avenue" }]}

- **Note**: Returns maximum 10 randomly selected upcoming events for carousel

---

GET `/api/events/:id`

Purpose: Get single event details

- **Auth Required**: Yes
- **URL Parameters**: `id` (event ObjectId)

Success Response (200):
{ "_id": "ObjectId", "title": "Jazz Night at The Spotted Cat", "description": "Live jazz music every Thursday night...", "date": "2025-10-15T00:00:00.000Z", "time": "8:00 PM", "location": "The Spotted Cat, Frenchmen Street", "category": { "_id": "...", "name": "Live Music", "slug": "live-music", "color": "#FF6B6B" }, "imageUrl": "https://...", "source": "user", "sourceUrl": null, "creator": { "_id": "...", "username": "jazzfan123" }, "likesCount": 15, "isLikedByUser": false, "isCreator": true, "createdAt": "2025-09-01T12:00:00.000Z"}

- **Error Response (404)**: Event not found

- **Error Response (401)**: Not authenticated

POST `/api/events`

Purpose: Create a new user event

- **Auth Required**: Yes

Request Body:
{ "title": "string (required, max 200 chars)", "description": "string (required, max 2000 chars)", "date": "ISO date string (required, must be future date)", "time": "string (optional, e.g., '7:00 PM')", "location": "string (required)", "category": "ObjectId (required, valid category ID)", "imageUrl": "string (optional, Cloudinary URL from upload)"}

Success Response (201):
{ "message": "Event created successfully", "event": { "_id": "ObjectId", "title": "My Awesome Event", "description": "...", "date": "2025-11-01T00:00:00.000Z", "time": "6:00 PM", "location": "City Park", "category": "ObjectId", "imageUrl": "https://...", "source": "user", "creator": "ObjectId", "likes": [], "likesCount": 0, "status": "upcoming", "createdAt": "2025-10-06T14:30:00.000Z" }}

- **Error Response (400)**: Validation errors (missing required fields, invalid date format, date in past)
- **Error Response (401)**: Not authenticated
- **Side Effects**: Adds event ID to user's `createdEvents` array

PUT `/api/events/:id`

Purpose: Update an existing user-created event

- **Auth Required**: Yes
- **Authorization**: Must be event creator
- **URL Parameters**: `id` (event ObjectId)

Request Body: (all fields optional, only send what needs updating)
{ "title": "string", "description": "string", "date": "ISO date string", "time": "string", "location": "string", "category": "ObjectId", "imageUrl": "string"}

Success Response (200):
{ "message": "Event updated successfully", "event": { /* updated event object */ }}

- **Error Response (400)**: Validation errors
- **Error Response (403)**: Not authorized (not event creator OR event from external source)

- **Error Response (404)**: Event not found
- **Note**: Cannot edit events with `source: 'eventbrite'` or `source: 'ticketmaster'`

DELETE `/api/events/:id`

Purpose: Delete a user-created event

- **Auth Required**: Yes
- **Authorization**: Must be event creator
- **URL Parameters**: `id` (event ObjectId)

Success Response (200):
{ "message": "Event deleted successfully" }

- **Error Response (403)**: Not authorized to delete this event
- **Error Response (404)**: Event not found
- **Side Effects**:
    - Removes event ID from creator's `createdEvents` array
    - Removes event ID from all users' `likedEvents` arrays

POST `/api/events/:id/like`

Purpose: Toggle like on an event (like if not liked, unlike if already liked)

- **Auth Required**: Yes
- **URL Parameters**: `id` (event ObjectId)
- **Request Body**: None

Success Response (200):
{ "message": "Event liked" | "Event unliked", "isLiked": true | false, "likesCount": 16}

- **Error Response (404)**: Event not found
- **Error Response (401)**: Not authenticated
- **Side Effects**:
    - **Like**: Adds user ID to event's `likes` array, increments `likesCount`, adds event ID to user's `likedEvents` array

- **Unlike**: Removes user ID from event's `likes` array, decrements `likesCount`, removes event ID from user's `likedEvents` array

---

POST `/api/events/sync`

Purpose: Manually trigger external API sync (admin/cron only)

- **Auth Required**: Yes
- **Authorization**: Admin role required (or valid cron secret)

Request Body:
{ "source": "eventbrite" | "ticketmaster" | "all" }

Success Response (200):
{ "message": "Sync completed successfully", "stats": { "added": 25, "updated": 5, "skipped": 10, "errors": 0 }}

- **Error Response (403)**: Not authorized (not admin)
- **Error Response (500)**: Sync failed with error details
- **Note**: Typically called by Vercel Cron job daily, but manual trigger available for admins

---

3. Users Routes

GET `/api/users/me`

Purpose: Get current authenticated user's profile

- **Auth Required**: Yes

Success Response (200):
{ "user": { "_id": "ObjectId", "username": "jazzlover22", "email": "user@example.com", "isVerified": true, "createdAt": "2025-08-15T10:00:00.000Z", "eventsCreatedCount": 5, "eventsLikedCount": 23 }}

- **Error Response (401)**: Not authenticated

GET `/api/users/:id/events`

Purpose: Get all events created by a specific user

- **Auth Required**: Yes
- **URL Parameters**: `id` (user ObjectId)

- **Query Parameters**:
    - `status`: "upcoming" | "passed" (default: "upcoming")
    - `limit`: number (default: 50)

Success Response (200):
{ "events": [/* array of event objects */], "total": 5}

- **Error Response (404)**: User not found
- **Note**: Useful for "My Events" tab in User Dashboard

GET `/api/users/:id/likes`

Purpose: Get all events liked by a specific user

- **Auth Required**: Yes
- **URL Parameters**: `id` (user ObjectId)
- **Query Parameters**:
    - `status`: "upcoming" | "passed" (default: "upcoming")
    - `limit`: number (default: 50)

Success Response (200):
{ "events": [/* array of event objects */], "total": 23}

- **Error Response (404)**: User not found
- **Note**: Useful for "Liked Events" tab in User Dashboard

---

4. Categories Routes

GET `/api/categories`

Purpose: Get all event categories (for filters, dropdowns)

- **Auth Required**: No (public endpoint)

Success Response (200):
{ "categories": [{ "_id": "ObjectId", "name": "Live Music", "slug": "live-music", "color": "#FF6B6B" }, { "_id": "ObjectId", "name": "Food & Drink", "slug": "food-drink", "color": "#4ECDC4" }]}

- **Note**: Small collection (~6-10 items), can be cached on frontend

---

5. Upload Route

POST `/api/upload`

Purpose: Upload image to Cloudinary

- **Auth Required**: Yes
- **Content-Type**: `multipart/form-data`
- **Request Body**: FormData with `image` field containing file

Success Response (200):
{ "imageUrl": "https://res.cloudinary.com/your-cloud/image/upload/v1234/local-pulse/events/abc123.jpg"}

- **Error Response (400)**: No file provided or invalid file type (must be image/jpeg, image/png, image/webp)
- **Error Response (413)**: File too large (max 5MB)
- **Error Response (500)**: Cloudinary upload failed
- **Note**: Returns Cloudinary URL to be included in event creation/update request

---

Error Response Format (Standardized)

All error responses follow this structure:

```
{
  "error": "Human-readable error message",
  "code": "ERROR_CODE_CONSTANT",
  "details": { /* optional, for validation errors */ }
}
```

Example validation error (400):

```
{
  "error": "Validation failed",
  "code": "VALIDATION_ERROR",
  "details": {
    "title": "Title is required",
```

```
    "date": "Date must be in the future"
  }
}
```

HTTP Status Codes Used

- **200**: Success (GET, PUT, DELETE, POST for non-creation actions like login)
- **201**: Created (POST for creating new resources like signup, create event)
- **400**: Bad Request (validation errors, malformed requests)
- **401**: Unauthorized (not authenticated, missing/invalid token)
- **403**: Forbidden (authenticated but not authorized for this action)
- **404**: Not Found (resource doesn't exist)
- **409**: Conflict (duplicate resource, e.g., username already taken)
- **413**: Payload Too Large (file upload exceeds size limit)
- **500**: Internal Server Error (unexpected server errors)

Authentication Flow

Protected endpoints require JWT in header:

Authorization: Bearer <JWT_TOKEN>

If token is missing, invalid, or expired:

```
{
  "error": "Authentication required. Please log in.",
  "code": "UNAUTHORIZED"
}
```

HTTP Status: 401

Frontend action: Redirect to login page, clear local auth state

---

Notes & Future Considerations

1. **Rate Limiting**: Not implemented in MVP, but consider adding:
    - 100 requests/minute per IP for public endpoints
    - 500 requests/minute per user for authenticated endpoints
2. **API Versioning**: Currently implicit v1 at `/api/`. If breaking changes needed, use `/api/v2/`
3. **Pagination**: Implemented via `limit` and `skip` query params. Consider cursor-based pagination for very large datasets in future.
4. **Caching**: Categories endpoint is good candidate for caching. Consider Redis or in-memory cache for featured events.
5. **Webhooks** (stretch goal): External APIs could notify via webhooks when events update, eliminating need for daily sync.
6. **GraphQL** (future consideration): If frontend needs become very complex with nested data requirements, consider GraphQL migration.